الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكوين والتعليم المهنيين
Ministère de la Formation et de l'Enseignement Professionnels

Centre National de l'Enseignement Professionnel à Distance (CNEPD)

المركز الوطني للتعليم المهني عن بعد (م.و.ت.م.ب)

COURS DE SYSTÈME D'EXPLOITATION

SÉRIE 03

OBJECTIF PÉDAGOGIQUE :

À l'issue de cette série, le stagiaire doit être capable de **c**onnaître :

1- Notion d'ordonnancement des processus ;
2- Les différentes politiques d'un ordonnanceur.

PLAN DE LA LEÇON :

I- NOTION DU PROCESSEUR :

Un processeur est un circuit intégré complexe caractérisé par une très grande intégration et doté des facultés d'interprétation et d'exécution des instructions d'un programme. Il est chargé d'organiser les tâches précisées par le programme et d'assurer leur exécution. Il doit aussi prendre en compte les informations extérieures au système et assurer leur traitement. C'est le cerveau du système.

II- L'ORDONNANCEMENT (LE SCHEDULING) :

Pour la très grande majorité des ordinateurs, avoir un seul processeur implique de ne pouvoir effectuer qu'un traitement à la fois. Or, à un instant donné, il est possible qu'il y ait plus de processus à exécuter qu'il n'y a de processeurs.

- On appelle **ordonnancement** ou **Scheduling**, l'organisation qui prend en charge l'allocation du processeur central aux programmes.
- On appelle **ordonnanceur** ou **Scheduler** la partie du système d'exploitation qui s'occupe de cette organisation et repartit le temps processeur entre ces programmes.
- Un ordonnanceur fait face à deux problèmes principaux :
  Le choix du processus à exécuter, et
  Le temps d'allocation du processeur au processus choisi.

1- Objectifs :

La technique d'un ordonnanceur doit viser les objectifs suivant :

L'équité : L'ordonnanceur doit servir les programmes d'une manière juste et équitable.

Rendement : L'ordonnanceur doit permettre et assurer l'exécution du maximum de programme (augmenter le rendement).

Utilisation des ressources : L'ordonnanceur doit assurer une occupation maximale des ressources.

2- Critères :

L'objectif d'une politique d'ordonnancement consiste à identifier le processus qui conduira à la meilleure performance possible du système. Certes, il s'agit là d'une évaluation subjective dans laquelle entrent en compte différents critères à l'importance relative variable. La politique d'ordonnancement détermine l'importance de chaque critère. Un certain nombre d'algorithmes ont fait leur preuve dans la mise en œuvre d'une politique d'ordonnancement.

La liste qui suit passe en revue des critères d'ordonnancement fréquemment utilisés.

Utilisation de l'UC : Pourcentage de temps pendant lequel l'UC exécute un processus.

L'importance de ce critère varie généralement en fonction du degré de partage du système.

Utilisation répartie : Pourcentage du temps pendant lequel est utilisé l'ensemble des ressources (outre l'UC, mémoire, périphérique d'E/S…)

Débit : Nombre de processus pouvant être exécutés par le système sur une période de temps donnée.

Temps de rotation : durée moyenne qu'il faut pour qu'un processus s'exécute. Le temps de rotation d'un processus comprend tout le temps que celui-ci passe dans le système. Il est inversement proportionnel au débit.

Temps d'attente : durée moyenne qu'un processus passe à attendre. Mesurer la performance par le temps de rotation présente un inconvénient : Le temps de production du processus accroît le temps de rotation ; Le temps d'attente représente donc une mesure plus précise de la performance.

Temps de réponse : Temps moyen qu'il faut au système pour commencer à répondre aux entrées de l'utilisateur.

Équité : degré auquel tous les processus reçoivent une chance égale de s'exécuter.

Priorités : attribue un traitement préférentiel aux processus dont le niveau de priorité est supérieur.

3- Priorités :

La priorité d'un processus est une information qui permet de classer un processus parmi d'autre lors d'un choix, il existe deux types de priorités :

Priorité fixe : défini a priori une fois pour toute. Elle peut être en fonction du temps estimé du processus.

Priorité variable au cours du temps : c'est une priorité qui peut changer dans le temps en fonction du temps d'attente écoulé, ou du temps déjà eu, etc.

Exemple : un processus qui fait beaucoup d'E/S passe beaucoup de temps à attendre et donc il faut lui allouer le PC dès qu'il le demande.

4- Niveaux :

Il est possible de distinguer trois niveau d'ordonnanceurs : à long terme, à moyen terme et à court terme. Leurs principales fonctions sont les suivantes :

À long terme : L'ordonnanceur fait la sélection de programmes à admettre dans le système pour leur exécution. Les programmes admis deviennent des processus à l'état **prêt**. L'admission dépend de la capacité du système (degré de multiprogrammation) et du niveau de performance requis.

À moyen terme : Il fait la sélection de processus déjà admis à débarquer ou rembarquer sur la mémoire. Il effectue ses tâches de gestion en fonction du degré de multiprogrammation du système, et aussi des requêtes d'E/S des périphériques.

À court terme : L'ordonnanceur à court terme a comme tâche la gestion des processus prêts. Il sélectionne en fonction d'une certaine politique le prochain processus à exécuter. Il effectue aussi le changement de contexte des processus. Il peut implanter un :

- **Ordonnancement préemptif :**

Avec réquisition où l'Ordonnanceur peut interrompre un processus en cours d'exécution si un nouveau processus de priorité plus élevée est inséré dans la file des Prêts.

- **Ordonnancement coopératif (non préemptif) :**

Ordonnancement jusqu'à achèvement : le processus élu garde le contrôle jusqu'à épuisement du temps qui lui a été alloué même si des processus plus prioritaires ont atteint la liste des Prêts.

L'ordonnanceur est activé par un événement : interruption du temporisateur, interruption d'un périphérique, appel système ou signal.

III- POLITIQUES DE L'ORDONNANCEMENT :

1- La politique FIFO (First In First Out)

L'ordonnancement est fait dans l'ordre d'arrivée en gérant une file unique des processus sans priorité ni réquisition : chaque processus s'exécute jusqu'à son terme ; le processus élu est celui qui est en tête de liste des Prêts : le premier arrivé. Cet algorithme est facile à implanter, mais il est loin d'optimiser le temps de traitement moyen

Exemple :

Exemple : **N° JOB**	**Temps d'exécution estime**	**Temps d'arrivé**	**Temps début d'exécution**	**Temps fin d'exécution**	**Temps réponse**
1	2 h	10 h 00	10 h 00	12 h 00	2 h
2	1h	10 h 10	12 h 00	13 h 00	2 h 50
3	25mn	10 h 25	13 h 00	13 h 25	3 h

Remarque :

- Le 1er job arrivé dans le système est servi.
- C'est une technique (politique) de scheduling non préemptive (coopérative) qui désavantage les jobs courts.
- Non préemptive== non arrêté algorithme avec réquisition AAR

2- La politique SJF (Shortest Job First) :

L'ordonnancement par ordre inverse du temps d'exécution (supposé connu à l'avance) : lorsque plusieurs travaux d'égale importance se trouvent dans une file, l'Ordonnanceur élit le plus court d'abord (les travaux les plus cours étant en tête de la file des prêts).

Cet algorithme possède l'inconvénient de la nécessité de connaissance du temps de service à priori et le risque de privation des tâches les plus longues. Afin de résoudre ces problèmes on pourra attribuer aux travaux une priorité croissante avec leur temps de séjour dans la file (temps d'attente). À temps d'attente égale, le travail le plus court est prioritaire. Toutefois, cette solution nécessite le calcul des priorités périodiquement et un réarrangement de la FA.

Exemple :

N° Job	Temps d'Exécution Estime	Temps Arrives	Temps Début Exécution	Temps Fin Exécution	Temps Réponse
1	2 h	10 h 00	10 h 00	12 h 00	2 h
2	1h	10 h 10	12 h 25	13 h 25	03 h 15
3	25mn	10 h 25	12 h 00	12 h 25	2 h

3- La politique d'ordonnancement par tourniquet (Round Robin) :

Il s'agit d'un algorithme ancien, simple et fiable. Le processeur gère une liste circulaire de processus.

Chaque processus dispose d'un quantum de temps pendant lequel il est autorisé à s'exécuter. Si le processus actif se bloque ou s'achève avant la fin de son quantum, le processeur est immédiatement alloué à un autre processus. Si le quantum s'achève avant la fin du processus, le processeur est alloué au processus suivant dans la liste et le processus précédent se trouve ainsi en queue de liste.

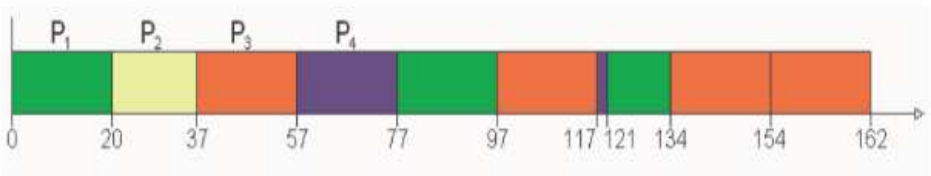
La commutation de processus dure un temps non nul pour la mise à jour des tables, la sauvegarde des registres. Un quantum trop petit provoque trop de commutations de processus et abaisse l'efficacité du processeur. Un quantum trop grand augmente le temps de réponse en mode interactif. On utilise souvent un quantum de l'ordre de 100 ms.

Exemple :

Etant donnes les processus suivants :

P1 d'une durée de 53 UT ; P2 d'une durée de 17 UT ;
P3 d'une durée de 68 UT ; P4 d'une durée de 24 UT

Nous paramétrons Q (quota) à 20 UT, nous obtenons le diagramme suivant :



Remarque :

- Temps de rotation et temps d'attente moyens
- Aucun processus n'est favorisé

4- La politique a plusieurs niveaux :

Les politiques présentées jusqu'à présent utilisent une seule file d'attente des processus prêts.

On choisit ici de définir plusieurs files de processus prêts, chaque file correspondant à un niveau de priorité ; on peut alors avoir n files de priorités différentes variant de 0 à n-1. Dans une file donnée, tous les

processus ont la même priorité et sont servis soit selon une politique à l'ancienneté sans préemption, soit selon une politique de tourniquet.

Le quantum peut être différent selon le niveau de priorité de la file. L'ordonnanceur sert d'abord tous les processus de la file de priorité n, puis ceux de priorité n-1 dès que la file de niveau n est vide et ainsi de suite...

On peut définir deux variantes de l'algorithme, fonction de l'évolution de la priorité des processus :

• Les priorités des processus sont constantes tout au long de leur exécution. À ce moment-là, un processus en fin de quantum est toujours réinséré dans la même file d'attente, celle correspondant à son niveau de priorité.

• Les priorités des processus évoluent dynamiquement en fonction du temps de service dont a bénéficié le processus. Ainsi un processus de priorité n, à la fin du quantum de la file n, s'il n'a pas terminé son exécution, n'est pas réinséré dans la file de priorité n, mais dans la file de priorité n-1. Et ainsi de suite... On cherche ici à minimiser les risques de famine pour les processus de faible priorité en faisant petit à petit baisser la priorité des processus de plus forte priorité.

IV- PERFORMANCE DES POLITIQUES D'ORDONNANCEMENT :

Les performances d'une politique pour un ensemble de processus donné peuvent être analysées si les informations appropriées relatives aux processus sont fournies.

Temps de rotation=Temps fin d'exécution - Temps d'arrivée

Temps d'attente=Temps de rotation – Durée d'exécution

$$\text{Temps moyen d'attente} = \frac{\sum Temps\ attente}{nbre\ de\ processus}$$

$$\text{Rendement} = \frac{\sum Temps\ d'exécution}{nbre\ de\ processus}$$

EXERCICE D'APPLICATION :

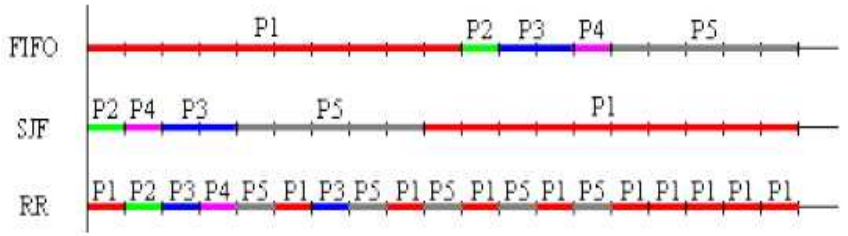
1- Citez les différentes politiques d'un ordonnanceur ?
2- Quelle est la différence entre un ordonnanceur préemptif et non préemptif ?
3- Cinq processus, P1, P2, P3, P4, P5 sont dans une file d'attente dans cet ordre (P1 est le premier, P5 est le dernier). Leur exécution demande un temps total de service exprimé en unités arbitraires :

Processus	P1	P2	P3	P4	P5
Temps de service estimé	10	1	2	1	5

1- Décrire l'exécution des processus dans le cadre des politiques d'ordonnancement FIFO, SJF, RR (avec un quantum de 1).

SOLUTION :

- Question 1 et 2 se référer au cours.
- Le schéma ci-dessous décrit l'enchaînement des processus :



	P1	**P2**	**P3**	**P4**	**P5**	**TOTAL**	**TEMPS MOYEN**
FIFO	10	1	13	14	19	57	11.4
SJF	19	1	4	2	9	35	7
RR	19	2	7	4	14	46	9.2

Le tableau, ci-dessus, indique les temps de présence, dans le système, des processus. La dernière colonne décrit le temps moyen passé par chaque processus. Il est clair que, sur cet exemple, **la stratégie SJF est la meilleure.**

RÉFÉRENCES BIBLIOGRAPHIQUES :

1. CT BULABULA DEMA Faustin, Cours de Systèmes d'exploitation et le Bureautique I, ISP/Bukavu en G1 IG, Inédit, 2004-2005
2. Ass TASHO KASONGO, Cours de l'Introduction à l'Informatique, ISP/Bukavu en G1 IG, Inédit, 2006-2007
3. Ass KYENDA SULIKA Dieu-donné, Cours de Systèmes d'exploitation, ISP/Bukavu en G1 IG, Inédit, 2000-2001.
4. AUMIAUX, M. Initiation à l'informatique de gestion, 2ème éd. Masson, Paris, 1983.
5. CAPRON, H.L. Computer, a tool for information age, 3ème ed. Upper saddle river, New Jersey, 2000.
6. CHAUMEL J.L.: L'implantation d'une technologie, nouvelle leçon corrigée, ed. Hériat, Montréal, 1989.
7. DANIEL, C. : Organiser le développement de la micro-informatique, Ed. d'organisations, Paris, 1987.
8. DELEPONE, J.F., Introduction théorique à l'informatique, Africa Computing, Abidjan, 2004.
9. DONALD H. : L'informatique : Un instrument de la gestion, Mc Graw-Hill, 1980.
10. DULONG, A. et SUTTER, E. : Les technologies de l'information, Paris, 1990,
11. JAVEAU, C. : Enquête par questionnaire, éd. ULB, 1995.
12. LOOIJEN M. : Information Systems : Management, Control and Maintenance, Kluwer Bedrijfsinformatique, 1998.

WEBO GRAPHIE

http://www.bestcours.com/systeme-exploitation
http://depinfo.u-cergy.fr/~pl/docs/slidesOS.pdf
http://deptinfo.cnam.fr/Enseignement/CycleA/AMSI/transparents_systemes/14_gestion_memoire.pdf